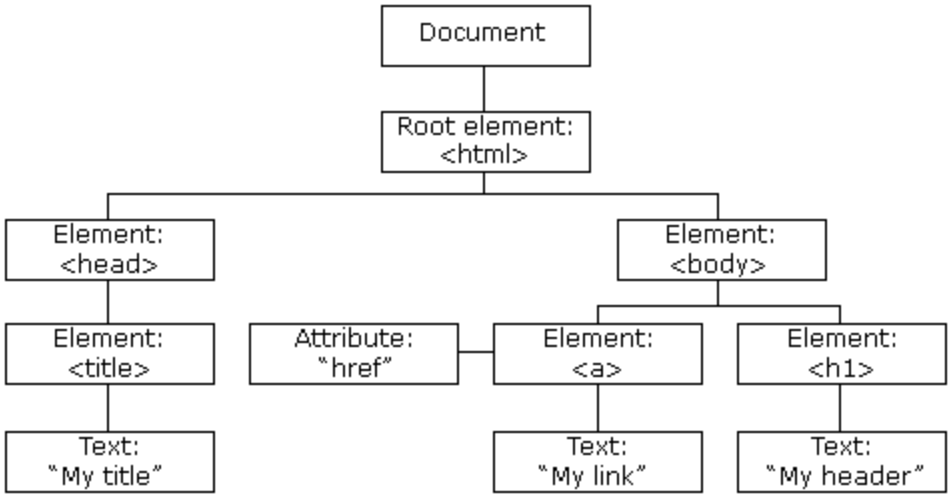
Date=26/10/2020
Lecture By=Manish Mahant
Subject ⇒Dom Manipulation Revision

IN PREVIOUS LECTURE **(QUICK RECAP) Date-23/10/2020**	**In Today's Lecture (Overview)**
AJAX Introduction jQuery ajax() Method Example Definition and Usage Syntax Questions For Self Practice // Assignment/ CC For the Day	The HTML DOM (Document Object Model) What is the DOM? JavaScript - HTML DOM Methods HTML DOM appendChild() Method Window setTimeout() Method Window setInterval() Method Questions for self practice / Assignment/CC For The Day

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a Document Object Model of the page.

The HTML DOM model is constructed as a tree of Objects:



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents
-

JavaScript - HTML DOM Methods

The DOM Programming Interface

The HTML DOM can be accessed with JavaScript (and with other programming languages).

In the DOM, all HTML elements are defined as objects.

The programming interface is the properties and methods of each object.

A property is a value that you can get or set (like changing the content of an HTML element).

A method is an action you can do (like add or deleting an HTML element).

Example

The following example changes the content (the `innerHTML`) of the `<p>` element with `id="demo"`:

Example

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

The getElementById Method

The most common way to access an HTML element is to use the `id` of the element.

In the example above the `getElementById` method used `id="demo"` to find the element.

The innerHTML Property

The easiest way to get the content of an element is by using the `innerHTML` property.

The `innerHTML` property is useful for getting or replacing the content of HTML elements.

The HTML DOM Document Object

The document object represents your web page.

If you want to access any element in an HTML page, you always start with accessing the document object.

Below are some examples of how you can use the document object to access and manipulate HTML.

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id

document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Property	**Description**
element.innerHTML = *new html content*	Change the inner HTML of an element
element.*attribute* = *new value*	Change the attribute value of an HTML element
element.style.*property* = *new style*	Change the style of an HTML element

Method	**Description**
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Adding Events Handlers

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler code to an onclick event

Finding HTML Elements

Often, with JavaScript, you want to manipulate HTML elements.

To do so, you have to find the elements first. There are several ways to do this:

- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name
- Finding HTML elements by CSS selectors
- Finding HTML elements by HTML object collections

Finding HTML Element by Id

The easiest way to find an HTML element in the DOM, is by using the element id.

This example finds the element with `id="intro"`:

Example

```
var myElement = document.getElementById("intro");
```

If the element is found, the method will return the element as an object (in myElement).

If the element is not found, myElement will contain `null`.

Finding HTML Elements by Tag Name

This example finds all `<p>` elements:

Example

```
var x = document.getElementsByTagName("p");
```

This example finds the element with `id="main"`, and then finds all `<p>` elements inside `"main"`:

Example

```
var x = document.getElementById("main");

var y = x.getElementsByTagName("p");
```

Changing HTML Content

The easiest way to modify the content of an HTML element is by using the `innerHTML` property.

To change the content of an HTML element, use this syntax:

```
document.getElementById(id).innerHTML = new HTML
```

This example changes the content of a `<p>` element:

Example

```
<html>

<body>

<p id="p1">Hello World!</p>

<script>

document.getElementById("p1").innerHTML = "New text!";

</script>

</body>
```

```
</html>
```

Example explained:

- The HTML document above contains a `<p>` element with `id="p1"`
- We use the HTML DOM to get the element with `id="p1"`
- A JavaScript changes the content (`innerHTML`) of that element to "New text!"

HTML Event Attributes

To assign events to HTML elements you can use event attributes.

Example

Assign an onclick event to a button element:

```
<button onclick="displayDate()">Try it</button>
```

HTML DOM appendChild() Method

Example

Append an item in a list:

```
var node = document.createElement("LI");                 // Create
a <li> node
var textnode = document.createTextNode("Water");         // Create
a text node
node.appendChild(textnode);                              // Append
the text to <li>
document.getElementById("myList").appendChild(node);     // Append
<li> to <ul> with id="myList"
```

Definition and Usage

The appendChild() method appends a node as the last child of a node.

Tip: If you want to create a new paragraph, with text, remember to create the text as a Text node which you append to the paragraph, *then* append the paragraph to the document.

You can also use this method to move an element from one element to another (See "More Examples").

Window setTimeout() Method

Example

Display an alert box after 3 seconds (3000 milliseconds):

```
setTimeout(function(){ alert("Hello"); }, 3000);
```

Definition and Usage

The setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds.

Tip: 1000 ms = 1 second.

Tip: The function is only executed once. If you need to repeat execution, use the setInterval() method.

Tip: Use the clearTimeout() method to prevent the function from running.

Syntax

```
setTimeout(function, milliseconds, param1, param2, ...)
```

Example

You can also refer to "named" function; Display an alert box after 3 seconds (3000 milliseconds):

```
var myVar;

function myFunction() {
  myVar = setTimeout(alertFunc, 3000);
}
```

```
function alertFunc() {
  alert("Hello!");
}
```

Example

Display a timed text:

```
var x = document.getElementById("txt");
setTimeout(function(){ x.value = "2 seconds" }, 2000);
setTimeout(function(){ x.value = "4 seconds" }, 4000);
setTimeout(function(){ x.value = "6 seconds" }, 6000);
```

Window setInterval() Method

Example

Alert "Hello" every 3 seconds (3000 milliseconds):

```
setInterval(function(){ alert("Hello"); }, 3000);
```

Definition and Usage

The setInterval() method calls a function or evaluates an expression at specified intervals (in milliseconds).

The setInterval() method will continue calling the function until clearInterval() is called, or the window is closed.

The ID value returned by setInterval() is used as the parameter for the clearInterval() method.

Tip: 1000 ms = 1 second.

Tip: To execute a function only once, after a specified number of milliseconds, use the setTimeout() method.

Syntax

```
setInterval(function, milliseconds, param1, param2, ...)
```

Parameter Values

Parameter	Description
function	Required. The function that will be executed
milliseconds	Required. The intervals (in milliseconds) on how often to execute the code. If the value is less than 10, the value 10 is used
param1, param2, ...	Optional. Additional parameters to pass to the *function* (Not supported in IE9 and earlier)

Example

You can also refer to a "named" function; Alert "Hello" every 3 seconds (3000 milliseconds):

```
var myVar;

function myFunction() {
  myVar = setInterval(alertFunc, 3000);
}

function alertFunc() {
  alert("Hello!");
}
```

Example

Display the current time (the setInterval() method will execute the function once every 1 second, just like a digital watch):

```
var myVar = setInterval(myTimer, 1000);
```

```
function myTimer() {
  var d = new Date();
  var t = d.toLocaleTimeString();
  document.getElementById("demo").innerHTML = t;
}
```

Questions for self practice / Assignment/CC For The Day

https://au-assignment.s3.ap-south-1.amazonaws.com/Week_18_Day_1_Challenge-01423459-4e60-41cf-abe3-e4ce57dbb7cb.pdf